NIKHEF-99-034
Revised 24-05-00

Fast access to parton densities, errors and correlations

EPDFLIB version 2.0

M. Botje*
NIKHEF, PO Box 41882, 1009DB Amsterdam, the Netherlands

May 24, 2000

Abstract

We describe a Fortran program which reads the input files of the MBFIT parton distribution set. Tools are provided to propagate the experimental statistical and systematic errors to any function of the parton densities. Also available are the strong coupling constant $\alpha_s(Q^2)$ and the results from a varity of systematic fits.

1 Introduction

In [1] we have presented a NLO QCD analysis of HERA and fixed target structure function data. The parton densities from this analysis are stored on HBOOK ntuple files. They cover a kinematic range of $9 \times 10^{-4} < x < 1$ and $1 < Q^2 < 9 \times 10^4$ GeV^2. Also available are the results from a series of systematic fits where the input parameters of the analysis (for instance α_s) were changed.

In this write-up we describe a Fortran program (EPDFLIB)[1] which gives fast access to these results and provides tools to propagate the statistical and systematic errors to any function of the parton densities.

On the web site `http://www.nikhef.nl/user/h24/qcdnum` the following files are stored:

```
epdflib2.f ............  Fortran source code.
epdflib2.inc ..........  Include file.

durat2.f ..............  Calculate d/u using epdflib.
```

*email: m.botje@nikhef.nl

[1] The differences between version 2 and 1 are given in appendix A.

```
fnc2.f ................  e-p NC structure functions.
fcc2.f ................  e-p CC structure functions.
fint2.f ...............  Sumrules, integrals and differentials.
astab2.f ..............  Alpha_s versus Q2.

mbfit1m.pdfnt2 ........  HBOOK ntuple file containing the parton
                         momentum densities (0.5 Mbyte).
mbfit1m.stfnt2 ........  Contains, for each quark flavor, the
                         contribution to the structure functions F2,
                         FL and xF3, calculated in NLO (1.5 Mbyte).
```

In case the transfer of the binary `.pdfnt2` and `.stfnt2` files fails, they can also be obtained through anonymous (binary) ftp:

```
>    ftp ftp.nikhef.nl
ftp> login:  anonymous
ftp> passwd: xx@yy.zz (your email address)
ftp> cd /pub/experiments/zeus/h24
ftp> bin
ftp> get mbfit1m.pdfnt2
ftp> get mbfit1m.stfnt2
ftp> quit
>
```

2 Error propagation

The input file to EPDFLIB contains the statistical and systematic covariance matrices of the fitted parameters and, as functions of x and Q^2, the central values of all parton momentum densities (or structure functions) as well as the derivatives of these densities to the fitted parameters. From this information we can calculate the covariance matrix element V_{ij} between any distribution $f_i(x_1, Q_1^2)$ and $f_j(x_2, Q_2^2)$ through

$$V_{ij} = <\Delta f_i \Delta f_j> = \sum_\lambda \sum_\mu \frac{\partial f_i(x_1, Q_1^2)}{\partial p_\lambda} \, V_{\lambda\mu} \, \frac{\partial f_j(x_2, Q_2^2)}{\partial p_\mu} \tag{1}$$

where $V_{\lambda\mu} = <\Delta p_\lambda \Delta p_\mu>$ is the covariance matrix element of the fitted parameters p_λ and p_μ.[2]

A nice feature of EPDFLIB is that it allows fast access to *any* linear combination of the parton densities, e.g.,

$$\begin{array}{lll} f_1 = & \sum_i e_i^2 \, x(q_i + \bar{q}_i) & \text{proton density,} \\ f_2 = & x(u - \bar{u}) + x(d - \bar{d}) & \text{valence sum,} \\ f_3 = & 2x(\bar{d} + \bar{u} + \bar{s} + \bar{c} + \bar{b}) & \text{sea distribution, etc.} \end{array}$$

For each linear combination EPDFLIB constructs parameterizations of f_i and $\partial f_i / \partial p_\lambda$ all of which are then accessible as functions of x and Q^2. Once the information is read in from a

[2]The covariance matrix $V_{\lambda\mu}$ can be the statistical, the systematic or, if the total error is to be calculated, the sum of both covariance matrices.

given input file you can calculate on the fly (that is, inside a loop over x and Q^2) any covariance matrix element V_{ij} as defined in Eq. (1).

Let us give as an example the Fortran code to calculate the valence distributions $xu_v = x(u-\bar{u})$ and $xd_v = x(d-\bar{d})$. The program gives the values, the errors and also the covariance matrix of these two distributions.

```
      implicit double precision (a-h,o-z) !all floating point in dp

C--   HBOOK memory (epdflib input files are CWN ntuples)
      parameter (nwpawc = 1000000)
      common /pawc/ hmemor(nwpawc)

C--   Upval and  Dnval definitions (notice the PDG convention)
      dimension uv(-5:5), dv(-5:5)
C--              bb  cb  sb  ub  db   g   d   u   s   c   b
C--              -5  -4  -3  -2  -1   0   1   2   3   4   5
      data uv /   0., 0., 0.,-1., 0., 0., 0., 1., 0., 0., 0. /
      data dv /   0., 0., 0., 0.,-1., 0., 1., 0., 0., 0., 0. /

C--   HBOOK initialization
      call hlimit ( nwpawc )

C--   Open input ntuple file
      call ep_open ( 20, 'mbfit1m.pdfnt2', ierr )

C--   Read the up and dn valence in memory locations 1 and 2
      call ep_read ( 1, 0, 0, uv, ierr )
      call ep_read ( 2, 0, 0, dv, ierr )

C--   Request 1=stat, 2=syst, 3=stat+syst errors
      mode = 3

      x1 = 0.1
      q1 = 4.
      x2 = 0.05
      q2 = 1000.

C--   Get results and errors
      uval = ep_valu ( 1, x1, q1,       ierr )
      uerr = ep_errs ( 1, x1, q1, mode, ierr )
      dval = ep_valu ( 2, x2, q2,       ierr )
      derr = ep_errs ( 2, x2, q2, mode, ierr )

C--   Get the covariance matrix
      vuu  = ep_cvar ( 1,x1,q1, 1,x1,q1, mode, ierr )    ! = uerr^2
      vud  = ep_cvar ( 1,x1,q1, 2,x2,q2, mode, ierr )
      vdd  = ep_cvar ( 2,x2,q2, 2,x2,q2, mode, ierr )    ! = derr^2
```

```
C--   Voila.....
```

Error propagation with EPDFLIB is by no means restricted to the input distributions like xu_v or xd_v. As we will see, errors can be calculated for any function F of the parton densities. Notice in this respect that EPDFLIB gives access to the *full* error information, i.e. what is known to the original QCD fit program is also known to EPDFLIB. The calculations described above apply to the statistical and experimental systematic errors only. Errors due to a variation of the input parameters (like α_s) must be constructed, in one way or another, from the difference of the central values and the results of the systematic fits, see [1] for an *ad-hoc* procedure.

3 Input files

The input data to EPDFLIB are stored in HBOOK ntuple format (cwn). The main advantage of this is that the input files can be transferred across machines (with binary ftp) and also that they allow random access. The EPDFLIB program can thus directly pick up the required information without having to read in the whole file and search through the large amount of data. The disadvantage is, of course, that you must have HBOOK installed on your machine if you want to use EPDFLIB.

As mentioned in Section 1 the input to EPDFLIB is stored on two different files:

1. `mbfit1m.pdfnt2`: This file contains the parton momentum densities and all information to calculate the errors on these;
2. `mbfit1m.stfnt2`: This file contains the structure functions, that is, for each quark distribution is stored the contribution to F_2, F_L and xF_3 in NLO.[3] Taking the appropriate linear combinations this information allows you to calculate the structure functions and/or inclusive cross sections for any deep inelastic scattering process. Also this file contains the full error information.

In total there are 870 two-dimensional distributions stored on the `.pdfnt2` and `.stfnt2` files. A straight dump of the QCD fit results would have required about 40 Mbyte of storage. Considerable reduction (factor 20) is obtained by fitting each distribution to a two dimensional cubic spline function [3] and storing the spline parameters instead. A nice property of splines is that they are additive, i.e. a linear combination of spline functions can be represented by taking the linear combination of the spline parameters. The EPDFLIB program exploits this feature to construct a parameterization of any linear combination requested by the user. Another nice feature is that derivatives and integrals over x and Q^2 (or $\ln x$ and $\ln Q^2$) are easy to calculate.

The spline fits represent the QCD results to an absolute accuracy of typically better than 10^{-3}. The EPDFLIB program provides, for each input distribution, two measures of the numerical precision: let F_i be the QCD fit result on a given x–Q^2 grid point i and f_i be the corresponding spline approximation. Then ε_{avg} and $\varepsilon_{\max}$ are defined as

$$\varepsilon_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} |F_i - f_i| \tag{2}$$

[3]For F_L the expressions given in [2] were used.

$$\varepsilon_{\max} = \max_i |F_i - f_i|$$

where n is the total number of grid points.

4 The EPDFLIB program

The source code of EPDFLIB is stored on the files `epdflib.f` and `epdflib.inc`. The include file contains two parameters you may want to re-set:

```
parameter ( mset = 50  )
parameter ( mpdf = 700 )
```

where `mset` is the maximum number of user id's (see `ep_read`) and `mpdf` is the maximum number of distributions which can be stored in memory (one central value and perhaps up to 28 derivatives $\partial f / \partial p_\lambda$ for each identifier).

As mentioned in the previous sections EPDFLIB uses HBOOK ntuple files. The standard HBOOK initialization should be done in the application program, see the example code in Section 2 or the example programs `durat2.f` etc.

All floating point variables in EPDFLIB are in double precision. Care should be taken that actual values are passed in double precision format e.g.:

```
implicit double precision (a-h,o-z)

..

x  = 0.001
Q2 = 4.

value = ep_valu ( 1, x    , Q2  , ierr )     !OK
value = ep_valu ( 1, 1.D-3, 4.D0, ierr )     !OK
value = ep_valu ( 1, 1.E-3, 4.  , ierr )     !wrong
```

5 EPDFLIB subroutine calls

In this section we describe the EPDFLIB subroutine calls. For quick reference these are listed in Table 1. Input and output floating point variables are in double precision. Output variables are indicated with an asterisk.

call EP_STOP (istop, iprint)

The EPDFLIB program will always stop execution (and print a message) if a fatal error is encountered such as a failure to read the input file or a failure to find a particular distribution on the input file. Every routine which addresses an input distribution, like `ep_valu`, gives the following error return code:

Table 1: Subroutines in EPDFLIB (not listed are eight functions associated with `ep_allf`).

Subroutine or function	Description
`EP_STOP ( istop, iprint )`	Set stop/print flag
`EP_OPEN ( lun, 'fname', *ierr )`	Open input file
`EP_READ ( id, itype, ifit, coeff, *ierr )`	Read parton density
`EP_VALU ( id, x, Q2, *ierr )`	Calculate parton density
`EP_ERRS ( id, x, Q2, mode, *ierr )`	Calculate error
`EP_CVAR ( id1,x1,q1, id2,x2,q2, mode, *ierr )`	Calculate covariance
`EP_ALLF ( id, ider, x, Q2, *ierr )`	Calculate f or df/dp_λ
`EP_UFUN ( ufun, x, Q2, *ierr )`	Calculate user function
`EP_UERR ( ufun, x, Q2, mode, *ierr )`	Error on user function
`EP_UVAR ( afun,x1,q1, bfun,x2,q2, mode, *ierr )`	Calculate covariance
`EP_ALFA ( Q2, factor, *ierr )`	Calculate α_s
`EP_LIMS ( *xmi, *xma, *qmi, *qma )`	Give the x–Q^2 limits
`EP_PARS ( *p, *d1, *d2, *d3, *if, ni, *no )`	Give fit parameters
`EP_EAVG ( id, *emax, *ierr )`	Give numerical error

```
ierr = -4      No distribution available for identifier id;
     = -3      Fit parameter number (ipder) out of range [1-50];
     = -2      Error identifier (mode) out of range [1-3];
     = -1      x or Q2 below lower limit;
     =  0      OK;
     = +1      x or Q2 above upper limit.
```

When `istop = 0` the routine will return a null value upon error, if `istop = 1` (default) execution will stop. By default EPDFLIB runs quiet (`iprint = 0`). Setting `iprint = 1` or `2` will generate some debug printout. You can (re)set the stop/print levels at any time.

```
call EP_OPEN ( lun, 'filename', *ierr )
```

Opens the ntuple input file `filename` on logical unit number `lun`. This routine will also clear the memory,[4] read some information from the file and set up an internal list of pointers. The error flag `ierr` is set to a non-zero value if the input file cannot be opened or if it cannot be read (the program will then stop execution). The file can be closed by a simple Fortran close

```
close ( lun )
```

```
call EP_READ ( id, itype, ifit, coeff, *ierr )
```

Read a parton density or structure function and store it in memory location `id`.

[4] Notice that it is not possible to have parton densities from two or more different input files in memory.

- `id` (`[1-50]`). User defined identifier of the parton density or structure function;
- `itype` (`[0-10]`). The type of distribution:

```
itype =  0    Parton density;
      =  1    F2;
      =  2    FL;
      =  3    xF3;
      =  8    Higher twist correction on F2p and F2d (see below);
      =  9    Nuclear effect correction on F2d (see below);
      = 10    Nuclear effect correction on xF3(nu-Fe) (see below).
```

  The parton densities should be read from the `.pdfnt2` and all other types from the `.stfnt2` files;

- `ifit` (`[-8,8]`). The fit identifier, that is `0` for the central fit, `+1` (`-1`) for the first systematic fit in the upward (downward) direction, and so on. We will not describe these fits here (see [1] for details) but simply give the index numbers by which EPDFLIB addresses these results:

```
ifit =  0      Central values;
     = +1,-1   Change the strong coupling constant alpha_s;
     = +2,-2   Change the strange quark content in the proton;
     = +3,-3   Change the nuclear correction on deuteron data;
     = +4,-4   Change the nuclear correction on xF3(nu-Fe);
     = +5,-5   Change the charm threshold;
     = +6,-6   Change the renormalization scale;
     = +7,-7   Change the factorization scale;
     = +8,-8   Upper/lower limits (envelope) of 10 additional
               systematic checks.
```

- `coeff`: An array dimensioned in the calling routine as `coeff(-5:5)`. This array defines the linear combination of parton densities (or structure functions)[5] to be stored in memory location `id`,

$$f = \sum_{i=-5}^{5} C_i \, xq_i \tag{3}$$

  where the flavor index i is according to the PDG convention: $i = 0$ indicates the gluon density, $i = 1, 2 \ldots 5$ indicate the quarks $d, u, \ldots b$ whereas $i = -1, -2 \ldots -5$ indicate the corresponding anti-quarks. See the example code in Section 2. If the higher twist or nuclear corrections are read in (`itype` = `8`, `9` or `10`) then the values given in `coeff` are irrelevant.

The error flag `ierr` is set to a non-zero value if the requested information is not present on the input file (this is considered to be a fatal error, that is, execution will stop).

We make the following remarks:

[5]Notice that xF_3 should always be a *non-singlet* structure function, that is, $xF_3 = \sum C_i \, xF_{3,i}$ with $\sum C_i = 0$.

- Error information is only available for distributions obtained from the central fit (`ifit = 0`). Error calculations on the distributions from systematic fits are allowed but should be avoided since CPU time is wasted on yielding a null result;

- The higher twist correction H applies to the proton and deuteron F_2 structure functions as measured at HERA and by fixed target experiments:

$$F_2^{\rm meas}(x,Q^2) = F_2^{\rm QCD}(x,Q^2)\ H(x,Q^2) \tag{4}$$

  Notice that H is left free in the QCD fit and therefore has errors;

- The F_2^d and xF_3 nuclear corrections are given as the ratios

$$F_2^d(x,Q^2)/F_2^N(x,Q^2) \quad \text{and} \quad xF_3^{\nu Fe}(x,Q^2)/xF_3^{\nu N}(x,Q^2) \tag{5}$$

  of the structure function (per nucleon) measured on the nuclear target to that calculated for a free nucleon (average of proton and neutron). There are no errors associated with these nuclear corrections (except that they were set differently for the central fit and for the systematic fits `3` and `4`).

```
value = EP_VALU ( id, x, Q2, *ierr )
```

Calculate the value of the parton distribution or structure function $f(x,Q^2)$ stored in memory location `id`. The error flag is set as given in the description of `ep_stop`. This is also the case for the routines described below.

```
value = EP_ERRS ( id, x, Q2, mode, *ierr )
```

Calculate the statistical (`mode = 1`), the systematic (`mode = 2`) or the quadratic sum of both errors (`mode = 3`) of the distribution stored at memory location `id`.

```
value = EP_CVAR ( id1,x1,Q21, id2,x2,Q22, mode, *ierr )
```

Calculate the covariance matrix element as defined in Eq. (1) of the distribution `id1` (evaluated at `x1, Q21`) and `id2` (evaluated at `x2, Q22`). The setting of the input flag `mode` is as for `ep_errs`.

```
value = EP_ALLF ( id, ipder, x, Q2, *ierr )
```

Calculate the function value $f(x,Q^2)$ or the derivative $\partial f(x,Q^2)/\partial p_\lambda$ to any fit parameter p_λ of the distribution stored in memory location `id`.

- `id` (`[1-50]`). Identifier of the parton density or structure function (as defined in `ep_read`);

- ipder ([0-50]). Fit parameter index (λ). For ipder = 0 the function returns the central value (like ep_valu). If ipder is non-zero then the value of the derivative to the fit parameter is returned. This may be zero for all x and Q^2 if the distribution does not depend on p_λ.[6]

Associated with ep_allf are 8 additional functions which allow to calculate derivatives and integrals with respect to x and Q^2:

ep_allf(id, ipder, x, Q2, *ierr)........	$f(x, Q^2)$
ep_dfdx(id, ipder, x, Q2, *ierr)........	$\partial f(x, Q^2)/\partial x$
ep_dlnx(id, ipder, x, Q2, *ierr)........	$\partial f(x, Q^2)/\partial \ln x$
ep_dfdq(id, ipder, x, Q2, *ierr)........	$\partial f(x, Q^2)/\partial Q^2$
ep_dlnq(id, ipder, x, Q2, *ierr)........	$\partial f(x, Q^2)/\partial \ln Q^2$
ep_Ifdx(id, ipder, x, Q2, *ierr)........	$\int_{x_1}^{x} f(z, Q^2)\, dz$
ep_Ilnx(id, ipder, x, Q2, *ierr)........	$\int_{x_1}^{x} f(z, Q^2)\, d\ln z$
ep_Ifdq(id, ipder, x, Q2, *ierr)........	$\int_{Q_1^2}^{Q^2} f(x, t)\, dt$
ep_Ilnq(id, ipder, x, Q2, *ierr)........	$\int_{Q_1^2}^{Q^2} f(x, t)\, d\ln t$

Here the lower limits of the integrals (x_1 and Q_1^2) are those of the x–Q^2 region where the parton densities are defined (see ep_lims).

Notice that in case ipder = 0 we obtain for instance from ep_dfdx the value of

$$\frac{\partial f(x, Q^2)}{\partial x}.$$

When ipder is set to a non-zero value we get

$$\frac{\partial}{\partial p_\lambda}\left[\frac{\partial f(x, Q^2)}{\partial x}\right].$$

The nine functions listed above can be regarded as building blocks to define complicated functions of the input distributions. The EPDFLIB program enables you to perform error calculations on these as will be described below.

[6] See ep_pars on how many parameters there are and which parameters were left free in the QCD fit. Such information is in principle not needed: one can simply loop over 50 hypothetical fit parameters and let EPDFLIB take care of the rest.

```
value = EP_UFUN ( ufun, x, Q2, *ierr )
```

Calculate the value of a user defined function `ufun`. This function (F) must be declared external in the calling routine. It must have the following syntax

```
double precision function ufun ( ipder, x, Q2, ierr )
```

and should return the central value $F(x, Q^2)$ if `ipder = 0` or the derivative $\partial F(x, Q^2)/\partial p_\lambda$ if `ipder` is non-zero. The definition of the output error flag `ierr` is up to the user.

Let us assume that a given parton distribution xq is stored in memory location `id` and that we want to compute the integral

$$I(x, Q^2) = \int_x^1 zq(z, Q^2)\, dz.$$

The function `ufun` might then be coded as follows:

```
double precision function fint ( ipder, x, Q2, ierr )

implicit double precision (a-h,o-z)

common /pass/ id

a1 = ep_ifdx ( id, ipder, 1.D0, Q2, ierr )  ! Int_xmin^1
a2 = ep_ifdx ( id, ipder, x   , Q2, ierr )  ! Int_xmin^x

fint = a1-a2

return
end
```

The function `ep_ufun` simply returns the value of `ufun` with `ipder` set to zero. However, the following routine calculates the *error* on `ufun`.

```
value = EP_UERR ( ufun, x, Q2, mode, *ierr )
```

Calculate the error on the user defined function `ufun` (see above for the proper definition of `ufun`). The input flag `mode` can be set to `1, 2, 3` to evaluate the statistical, the systematic or the sum of both errors. Notice that `ep_uerr` assumes that the derivatives $\partial F/\partial p_\lambda$ to the fit parameters p_λ are correctly calculated in `ufun`. The evaluation of the central value is irrelevant as far as `ep_uerr` is concerned.

For instance, using the routine `fint` as given above, we find from `ep_uval` and `ep_uerr` the following results for the momentum fractions at $Q^2 = 4$ GeV^2 ($x\Sigma$ is the sum of all quark and anti-quark densities):[7]

$$\int_{0.001}^1 xg\, dx \qquad = \quad 0.393 \pm 0.018 \quad (\text{stat.} + \text{syst.})$$

[7]The values are slightly different from those given in [1] because there, by mistake, the lower limit was set to $x = 9 \times 10^{-4}$ instead of to $x = 10^{-3}$.

$$\begin{aligned}\int_{0.001}^{1} x\Sigma\, dx &= 0.594 \pm 0.018 \\ \int_{0.001}^{1} (xg + x\Sigma)\, dx &= 0.987 \pm 0.002\end{aligned}$$

where the error on the last integral is much smaller than that on the first two since the total momentum fraction was constrained to be unity in the QCD fit. This example clearly illustrates the importance of taking into account the correlations between the parton densities.

value = EP_UVAR (afun,x1,Q21, bfun,x2,Q22, mode, *ierr)

Calculate the covariance matrix element V_{ab} of the functions `afun` and `bfun`. This routine is the equivalent of `ep_cvar`.

call EP_ALFA (Q2, factor, *ierr)

Calculate the strong coupling constant α_s. The input value $\alpha_s(M_Z^2) = 0.118 \pm 0.005$, the quark masses and the order of the calculation (NLO) are taken from the input file.

- `Q2`: input value of Q^2;
- `factor`: offset the value of α_s by `factor` times the error. Thus we get $\alpha_s(M_Z^2) = (0.113, 0.118, 0.123)$ for `factor` = (`-1,0,1`);
- `ierr`: error return code

```
ierr = -3      No alpha_s information available;
     = -2      Input alpha_s negative or zero;
     = -1      Q2 below the value of lambda2;
     =  0      OK;
```

  For `ierr` = `-1`, the value of α_s is arbitrarily set to 100, for the other error conditions it is set to zero. Upon error EPDFLIB will stop execution provided `istop` is set to a non-zero value.

call EP_LIMS (*xmi, *xma, *qmi, *qma)

Give the limits of the spline parameterizations. To be called after `ep_open`.

call EP_PARS (*par, *d1, *d2, *d3, *ifit, nin, *npar)

Give information on the fitted parameters. To be called after `ep_open`.

- `par`. Array containing the parameter values on output;
- `d1`, `d2`, `d3`. Arrays containing the statistical, the systematic and the total error on the parameters;

- `ifit`. Integer array containing on output the value `ifit(i) = (0)1` if parameter (i) is (fixed) fitted;
- `nin`. Dimension of the above arrays as declared in the calling routine (50 or larger);
- `npar`. Total number of parameters (fixed + fitted).

A fatal error (stop execution) will occur if the dimension of `par` etc. is not large enough to hold the complete parameter list.

```
value = EP_EAVG ( id, *emax, *ierr )
```

Give the average and maximum numerical error on the distribution `id`. For each parton density stored on the input file these errors are defined in Eq. (2). For a linear combination (see Eq. (3) in `ep_read`) the numerical uncertainties are calculated from

$$
\begin{aligned}
\varepsilon_{\mathrm{avg}} &= \frac{1}{m} \sum_{i=-5}^{5} |C_i| \, \varepsilon^i_{\mathrm{avg}} \\
\varepsilon_{\mathrm{max}} &= \sum_{i=-5}^{5} |C_i| \, \varepsilon^i_{\mathrm{max}}
\end{aligned}
$$

where m is the number of distributions for which the coefficients C_i are non-zero. This routine is useful if you work at very large x ($>$ 0.8–0.9, say) where numerical errors might spoil the result.

6 Examples

In this section we give a few detailed examples to illustrate the error propagation with EPDFLIB. The error on a function F of the input distributions f_i is calculated in `ep_uerr` using Eq. (1). What we have to provide (in `ufun`) are the derivatives $\partial F/\partial p_\lambda$ with respect to the fitted parameters p_λ. These have to be written, in one way or another, as a function of the f_i and the $\partial f_i/\partial p_\lambda$. Here it is important to clearly distinguish two cases:

- The function F is linear in the input distributions f_i. In this case we have the following property for the derivative of F

$$
\frac{\partial}{\partial p_\lambda} F(f_1, f_2, \ldots) = F\left(\frac{\partial f_1}{\partial p_\lambda}, \frac{\partial f_2}{\partial p_\lambda}, \ldots\right), \tag{6}
$$

  that is, we replace the f_i by the $\partial f_i/\partial p_\lambda$ to calculate the derivative. The above is true, of course, for linear combinations of the f_i with coefficients which may or may not depend on x and Q^2 and also for linear operations like differentiation or integration with respect to x or Q^2.[8] Convolution integrals with splitting or coefficient functions are also linear operations. Notice that in case of deep inelastic scattering all structure functions and cross sections are linear functions of the parton densities.

[8] A linear combination with constant coefficients should be defined with `ep_read` after which the error is directly available using `ep_errs`.

- The function F is non-linear. Examples are a calculation of ratios like d/u and F_2^n/F_2^p or the multiplication of F_2^p (which has errors) by the higher twist correction (which also has errors). If F is the result of simple binary operations like multiplication or division it is easy to calculate $\partial F/\partial p_\lambda$ analytically. For instance the calculation of hadron-hadron cross sections (e.g. Drell-Yan) requires the evaluation of convolution integrals of the parton densities. Generically,
$$\sigma = \sum_{ij} f_i \otimes f_j \otimes \hat{\sigma}_{ij}$$
so that we obtain for the derivative with respect to p_λ:
$$\frac{\partial \sigma}{\partial p_\lambda} = \sum_{ij} \left[\frac{\partial f_i}{\partial p_\lambda} \otimes f_j + f_i \otimes \frac{\partial f_j}{\partial p_\lambda} \right] \otimes \hat{\sigma}_{ij}. \tag{7}$$
If F is a very complicated function of the parton densities we have to resort to a numerical calculation of $\partial F/\partial p_\lambda$ (see below).

As an example of a linear function let us calculate the neutral current (NC) 'reduced' cross section for deep inelastic ep scattering at HERA which for not too high Q^2 (< 1000 GeV^2) can be written as
$$\sigma^{\rm NC}(x,Q^2) = F_2^{\rm NC}(x,Q^2) - \frac{y^2}{Y_+} F_L^{\rm NC}(x,Q^2) \tag{8}$$
where $y = Q^2/(xs)$, $Y_+ = 1 + (1+y)^2$ and $\sqrt{s}$ is the center-of-mass energy. Our main code looks as follows:

```
      ..

      external signc

      common /pass/ ecom2,idf2,idfl

C--   Proton definition
      dimension pr(-5:5)
C--               bb  cb  sb  ub  db   g   d   u   s   c   b
      data pr /   1., 4., 1., 4., 1., 0., 1., 4., 1., 4., 1. /

      do i = -5,5
        pr(i) = pr(i)/9.
      enddo

      idf2 = 1
      idfl = 2

      call ep_open ( lun, 'mbfit1m.stfnt2', ierr )
      call ep_read ( idf2, 1, 0, pr, ierr )   ! read F2(proton)
      call ep_read ( idfl, 2, 0, pr, ierr )   ! read FL(proton)

      ..
```

```
ecom2 = 4. * e_electron * e_proton
xsec  = ep_ufun ( signc, x, Q2, ierr )
xerr  = ep_uerr ( signc, x, Q2, mode, ierr )

..
```

The function signc which calculates the (double differential) NC cross section (and also the derivatives to the fitted parameters) can be coded as

```
double precision function signc ( ipder, x, q, ierr )

implicit double precision (a-h,o-z)

common /pass/ ecom2,idf2,idfl

..

F2 = ep_allf ( idf2, ipder, x, q, ierr )
FL = ep_allf ( idfl, ipder, x, q, ierr )

..

signc = F2 - (y*y)/yplus * FL

return
end
```

Notice that the calculation of a single differential cross section, which involves integration over x or Q^2 does not spoil the simplicity of the error calculation because integration is a linear operation. One just has to incorporate a numerical integration routine (like DGAUSS, CERN D103) in the function `signc`.

Next we give three methods to calculate the error on a non-linear function for which we chose the ratio $R = d/u$, see the example job `durat2.f`.

1. First of all we can calculate the covariance matrix of d and u with `ep_cvar`. Standard error propagation then gives

$$\frac{(\Delta R)^2}{R^2} = \frac{V_{dd}}{d^2} - 2\frac{V_{du}}{du} + \frac{V_{uu}}{u^2}. \tag{9}$$

2. Secondly we can analytically calculate in `ufun` the derivative of R with respect to p_λ:

$$\frac{\partial R}{\partial p_\lambda} = R\left[\frac{1}{d}\,\frac{\partial d}{\partial p_\lambda} - \frac{1}{u}\,\frac{\partial u}{\partial p_\lambda}\right]. \tag{10}$$

   The function `ufun` might be coded as follows:

```
double precision function rat ( ipder, x, q, ierr )

implicit double precision (a-h,o-z)

dn  = ep_allf ( 1, 0, x, q, ierr )
up  = ep_allf ( 2, 0, x, q, ierr )

rat = dn/up

if(ipder.eq.0) return

dd  = ep_allf ( 1, ipder, x, q, ierr )
du  = ep_allf ( 2, ipder, x, q, ierr )

rat = rat * ( dd/dn - du/up )

return
end
```

The error on d/u can now be calculated with `ep_uerr`.

3. Finally we can calculate in `ufun` the derivative $\partial R/\partial p_\lambda$ numerically. For any function $F(f_1, f_2, \ldots)$ this may be done as follows: define

$$f_i^\pm = f_i \pm \frac{\partial f_i}{\partial p_\lambda} \Delta p_\lambda. \tag{11}$$

An estimate for the derivative of F is then obtained from[9]

$$\frac{\partial F(f_1, f_2, \ldots)}{\partial p_\lambda} = \frac{1}{2\Delta p_\lambda} \left[F(f_1^+, f_2^+, \ldots) - F(f_1^-, f_2^-, \ldots) \right]. \tag{12}$$

Such a calculation should be fairly insensitive to the choice of Δp_λ, provided it is not taken to be 'too large' or 'too small'; we recommend to set it to the error on p_λ as given by `ep_pars`. We leave it for the reader to work out the function `ufun` for the d/u ratio (see `durat2.f`).

In some cases it is convenient to combine linear and non-linear error propagation. For instance if we are interested in the ratio of the charged current (CC) to NC cross sections at HERA we can write a user function `sigcc` in addition to `signc` given earlier in this section. The covariance matrix between `sigcc` and `signc` is obtained from calls to `ep_uvar`. The error on $\sigma^{\mathrm{CC}}/\sigma^{\mathrm{NC}}$ is then calculated from the standard error propagation formula for ratios as given in Eq. (9).

[9] Notice that we calculate the derivative from $\partial F/\partial p = (F^+ - F^-)/2\Delta p$, which is exact up to second order in p, instead of from $\partial F/\partial p = (F^+ - F)/\Delta p$, which is only accurate up to first order.

7 Summary

In this note we have described a Fortran program to read the parton momentum densities and structure functions obtained from the QCD fits given in ref. [1]. In addition to the central values the results of 7 systematic QCD fits and the upper and lower limits (envelope) of 10 additional systematic checks are available. The program calculates the errors on the central values from the statistical or the systematic covariance matrix of the fitted parameters. Tools are provided to propagate these errors to any function of the parton densities or structure functions.

We emphasize here that the errors from the QCD fits are not determined by the experimental uncertainties alone. They also depend, and maybe quite strongly, on the various assumptions made in the QCD analysis, in particular on the parameterizations chosen to describe the parton densities at the input scale Q_0^2. This explains the central rôle which is played in the error calculations by the covariance matrix of the fitted parameters and the derivatives of the parton densities, or structure functions, to these parameters.

A Differences between version 2 and 1

Here we give here a brief description of the differences between EPDFLIB version 2 and 1.

1. Version 2 is about a factor of 15 faster than version 1;

2. The spline parameterizations are now based on b-splines [3] instead of on the numerically less stable ‘one-sided’ representation. The version 1 input files can therefore not be read by version 2 and *vice versa*;

3. Version checking mechanism to protect against reading a wrong (outdated) input file;

4. Calculation of $\alpha_s(Q^2)$.

To change from version 1 to version 2 there is no need to modify your application program. You must, of course, link to version 2 and read the `.pdfnt2` or `.stfnt2` files in `ep_open`.

References

[1] M. Botje, ‘A QCD analysis of HERA and fixed target structure function data’, DESY-99-038, NIKHEF-99-011, hep-ph/9912439 (to appear in Eur. Phys. J. C).

[2] G. Altarelli and G. Martinelli, Phys. Lett. B76 (1978) 89.

[3] M. Botje, ‘Spline fits to one and two dimensional functions’, in preparation.